ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

Факультет безопасности информационных технологий

Кафедра проектирования и безопасности компьютерных систем

Дисциплина:

«Операционные системы»

ОТЧЕТ О ВЫПОЛНЕНИИ

ЛАБОРАТОРНОЙ РАБОТЫ №8

Выполнил:

Студент гр. N3247 Гаврилова В. В.

Проверил:

Ханов А. Р.

Санкт-Петербург

2021г.

Задание:

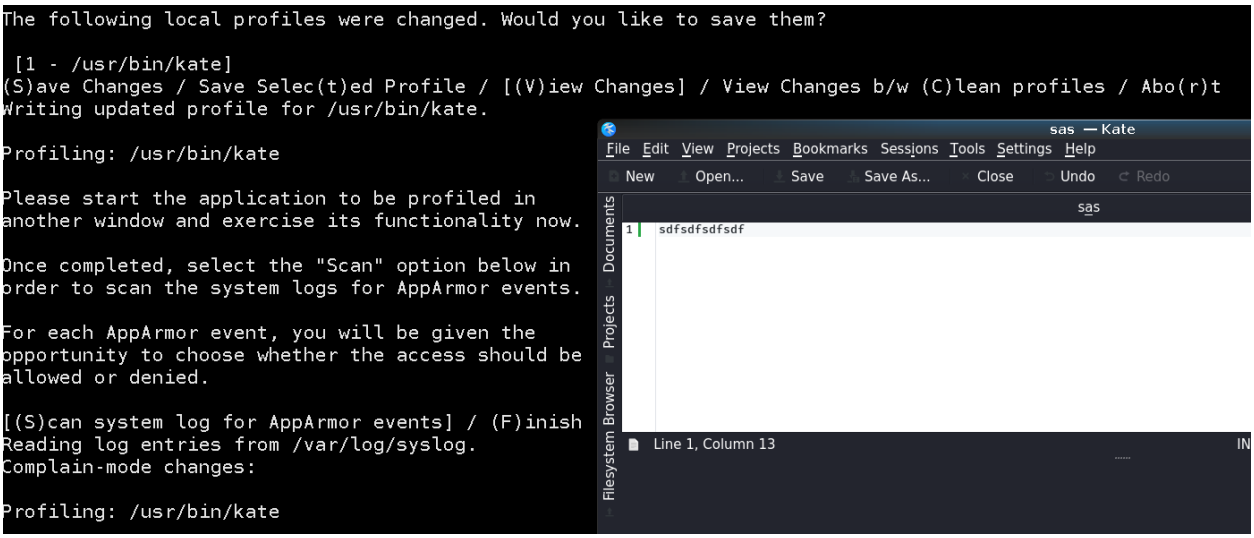
1. Настроить Apparmor для мониторинга сложного приложения и продемонстрировать его работу при ограниченных правах (оконное приложение или веб-сервер)
2. Настроить selinux в режиме мандатного доступа (CentOS и др.) и продемонстрировать работу в двухуровневой модели.
3. Написать собственный PAM модуль для аутентификации. (СЛОЖНЫЙ)

Задание 1. AppArmor

Какое простое оконное приложение первым приходит в голову? Лично мне блокнот kate. Возьмем kate и запрофилируем его. Для этого введем команды для создания файла профиля и его генерации, после чего в новой терминальной сессии запустим его и поделаем в kate что-нибудь.

Затем сканируем логи и разрешаем/запрещаем различные действия kate.
Теперь мы можем взглянуть на сгенерированный профиль:

```
apt-get update && apt-get install apparmor-utils
systemctl start apparmor.service
aa-autodep kate
aa-genprof kate
```



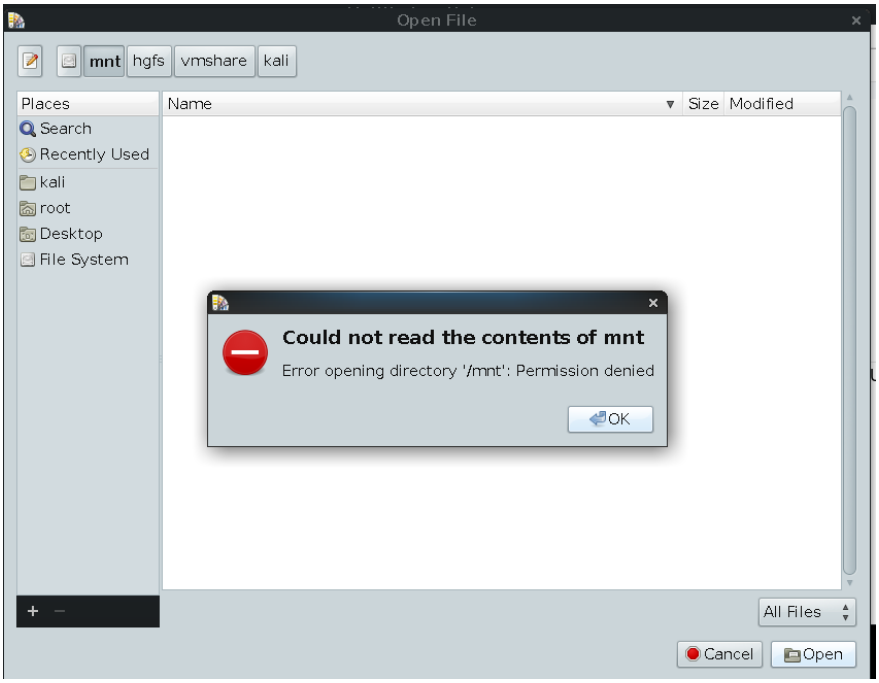
Посмотрим на сгенерированный профиль:

```
  GNU nano 5.2                                                  /etc/apparmor.d/usr.bin.kat
# Last Modified: Fri May 21 18:50:25 2021
#include <tunables/global>

/usr/bin/kate {
  #include <abstractions/base>
  #include <abstractions/lightdm>

  capability net_bind_service,
  capability sys_nice,

  signal send set=term peer=/usr/bin/kate//null-/usr/lib/x86_64-linux-gnu/libexec/kf5/kioslave5,

  owner /etc/fonts/** r,
  owner /mnt/hgfs/vmshare/kali/ r,
  owner /mnt/hgfs/vmshare/kali/.lol.txt.kate-swp w,
  owner /mnt/hgfs/vmshare/kali/.sas.kate-swp w,
  owner /mnt/hgfs/vmshare/kali/lol.txt rw,
  owner /mnt/hgfs/vmshare/kali/sas rw,
  owner /mnt/hgfs/vmshare/kali/test2 rw,
  owner /proc/*/mounts r,
  owner /proc/10625/cmdline r,
  owner /proc/sys/kernel/core_pattern r,
  owner /proc/sys/kernel/random/boot_id r,
  owner /root/ r,
  owner /root/.Xauthority r,
  owner /root/.cache/icon-cache.kcache rw,
  owner /root/.cache/ksycoca5_en_VD4iE6bBCAcOEGqobz+eaAH_TeI= r,
  owner /root/.config/#530453 rw,
  owner /root/.config/#530466 rw,
  owner /root/.config/#530467 rw,
  owner /root/.config/#530469 rw,
  owner /root/.config/QtProject.conf rw,
  owner /root/.config/QtProject.conf.lock rwk,
  owner /root/.config/QtProject.conf.ntybkw rwl,
  owner /root/.config/QtProject.conf.tRfOAp rwl,
  owner /root/.config/gtk-2.0/ r,
  owner /root/.config/gtk-2.0/gtkfilechooser.ini rw,
  owner /root/.config/gtk-2.0/gtkfilechooser.ini.QRLM30 rw,
  owner /root/.config/gtk-2.0/gtkfilechooser.ini.QVWW30 rw,
```

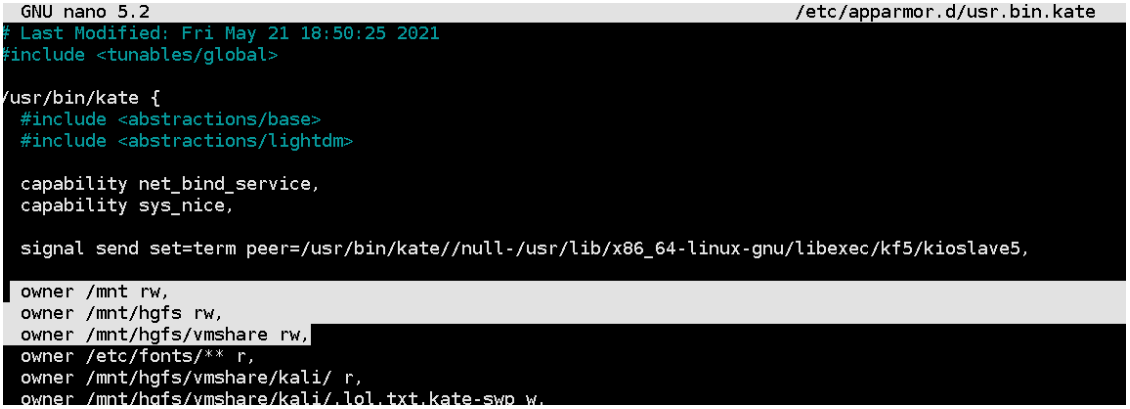
Давайте попробуем заэнфорсить профиль, посмотрим, как работает:

```
Finished generating profile for /usr/bin/kate.
root@kali:/mnt/hgfs/vmshare/kali# nano /etc/apparmor.d/usr.bin.kate
root@kali:/mnt/hgfs/vmshare/kali# aa-enforce kate
Setting /usr/bin/kate to enforce mode.
root@kali:/mnt/hgfs/vmshare/kali#
```

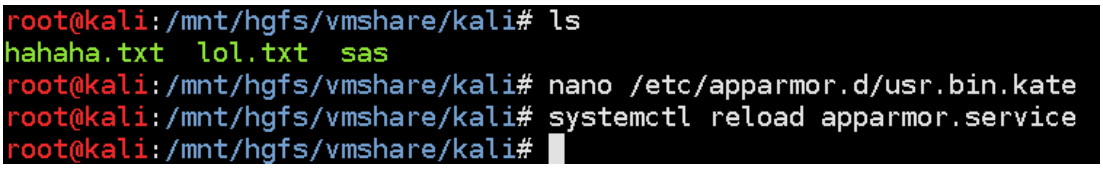
Вот незадача, kate не может посмотреть директорию /mnt :



А заодно с ней не дает посмотреть директории /hgfs и /vmshare, /kali однако дает смотреть, читать писать в файлы в ней.
Давайте это исправим, допишем разрешения на листинг и запись директорий:



Перезагрузим сервис apparmor:



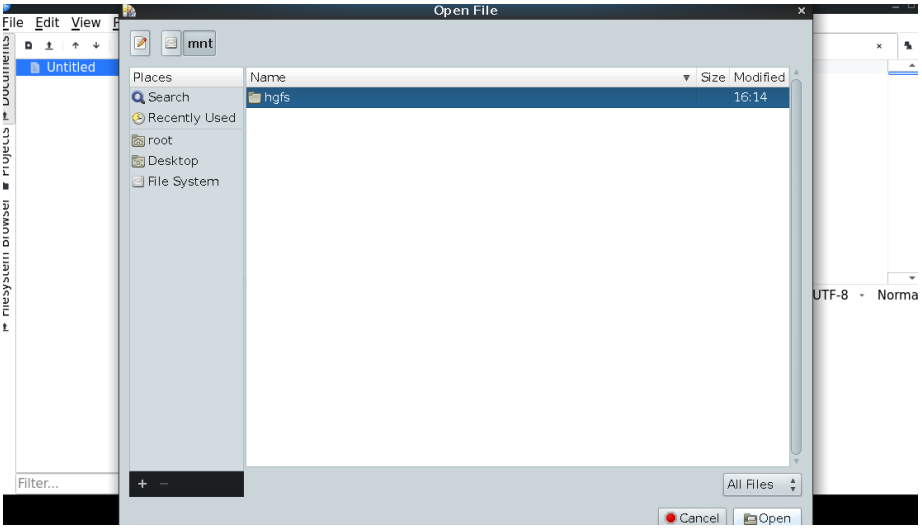
Однако это не привело к разрешению просмотра директорий. Да и вообще стоит сказать, что при выключении сервиса работающие профили остаются активными. Для того чтобы изменения вступили в силу, пришлось поломать голову. Помогло:

```
touch /etc/apparmor.d/disabled/usr.bin.kate
apparmor_parser -R /etc/apparmor.d/usr.bin.kate
reboot # Без ребута не работало
```

И заново:

```
systemctl start apparmor.service
aa-enforce kate
```

Вот результат:



Теперь мы при работающем профиле можем просматривать директории из kate.

Задание 2. SELinux

Посмотрим статус системы:

```
[root@localhost ~]# sestatus -v
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   permissive
Mode from config file:          permissive
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Max kernel policy version:      31

Process contexts:
Current context:                unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
Init context:                   system_u:system_r:init_t:s0
/usr/sbin/sshd                  system_u:system_r:sshd_t:s0-s0:c0.c1023

File contexts:
Controlling terminal:           unconfined_u:object_r:user_devpts_t:s0
/etc/passwd                     system_u:object_r:passwd_file_t:s0
/etc/shadow                     system_u:object_r:shadow_t:s0
/bin/bash                       system_u:object_r:shell_exec_t:s0
/bin/login                      system_u:object_r:login_exec_t:s0
/bin/sh                         system_u:object_r:bin_t:s0 -> system_u:object_r:shell_exec_t:s0
/sbin/agetty                    system_u:object_r:getty_exec_t:s0
/sbin/init                      system_u:object_r:bin_t:s0 -> system_u:object_r:init_exec_t:s0
/usr/sbin/sshd                  system_u:object_r:sshd_exec_t:s0
[root@localhost ~]#
```

В /etc/selinux/config пропишем тип mls (многоуровневая система безопасности):

```
Файл  Правка  Вид  Поиск  Терминал  Справка

# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=mls
```

После чего перевыставляем в соответствии с типом метки для файлов и директорий по всей ФС:

```
[root@localhost ~]# touch ./autorelabel
[root@localhost ~]# setenforce 0
[root@localhost ~]# getenforce
Permissive
[root@localhost ~]# reboot
```

```
[root@localhost ~]# sestatus | grep mls
Loaded policy name:             mls
[root@localhost ~]# dmesg | grep "SELinux is preventing"
[root@localhost ~]# setenforce 1
[root@localhost ~]# getenforce
Enforcing
[root@localhost ~]# reboot
```

Отлично, теперь MLS активна, посмотрим, что у нас по уровням и категориям доступа:

```
#
# Assumptions: using below MLS labels.
#  SystemLow
#  SystemHigh
#  Unclassified
#  Secret with compartments A and B.
#
# SystemLow and SystemHigh
s0=SystemLow
s15:c0.c1023=SystemHigh
s0-s15:c0.c1023=SystemLow-SystemHigh

# Unclassified level
s1=Unclassified

# Secret level with compartments
s2=Secret
s2:c0=A
s2:c1=B

# ranges for Unclassified
s0-s1=SystemLow-Unclassified
s1-s2=Unclassified-Secret
s1-s15:c0.c1023=Unclassified-SystemHigh

# ranges for Secret with compartments
s0-s2=SystemLow-Secret
s0-s2:c0=SystemLow-Secret:A
s0-s2:c1=SystemLow-Secret:B
s0-s2:c0,c1=SystemLow-Secret:AB
s1-s2:c0=Unclassified-Secret:A
s1-s2:c1=Unclassified-Secret:B
s1-s2:c0,c1=Unclassified-Secret:AB
s2-s2:c0=Secret-Secret:A
s2-s2:c1=Secret-Secret:B
s2-s2:c0,c1=Secret-Secret:AB
s2-s15:c0.c1023=Secret-SystemHigh
s2:c0-s2:c0,c1=Secret:A-Secret:AB
s2:c0-s15:c0.c1023=Secret:A-SystemHigh
s2:c1-s2:c0,c1=Secret:B-Secret:AB
s2:c1-s15:c0.c1023=Secret:B-SystemHigh
s2:c0,c1-s15:c0.c1023=Secret:AB-SystemHigh
```

MLS в SELinux по умолчанию реализовывает двухуровневую модель безопасности (2 уровня: s1- несекретно, s2 - секретно). Этого вполне хватит для демонстрации. Создам двух пользователей с соответствующими уровнями.

```
[root@localhost ~]# semanage user -m -r s1-s2 user_u
[root@localhost ~]# semanage login -a -s user_u -r s1 soldier
[root@localhost ~]# semanage login -a -s user_u -r s2 officer
[root@localhost ~]# semanage login -l

Имя входа            Пользователь SELinux Диапазон MLS/MCS     Служба

__default__          user_u               s0-s0                *
officer              user_u               s2                   *
root                 root                 s0-s15:c0.c1023      *
soldier              user_u               s1                   *
system_u             system_u             s0-s15:c0.c1023      *
[root@localhost ~]#
```

Кину на их домашние папки соответствующие уровни доступа:

```
[root@localhost home]# chcon -l s1 soldier
[root@localhost home]# chcon -l s2 officer
[root@localhost home]# ls -laZ
drwxr-xr-x. root      root      system_u:object_r:home_root_t:s0 .
dr-xr-xr-x. root      root      system_u:object_r:root_t:s0      ..
drwx------. officer   officer   user_u:object_r:user_home_dir_t:s2 officer
drwxr-xr-x. root      root      unconfined_u:object_r:user_home_dir_t:s0 solar
drwx------. soldier   soldier   user_u:object_r:user_home_dir_t:s1 soldier
drwx------. user      user      unconfined_u:object_r:user_home_dir_t:s0 user
drwx------. usertest1 usertest1 unconfined_u:object_r:user_home_dir_t:s0 usertest1
[root@localhost home]#
```

```
[root@localhost home]# setenforce 1
[root@localhost home]# getenforce
Enforcing
[root@localhost home]#
```

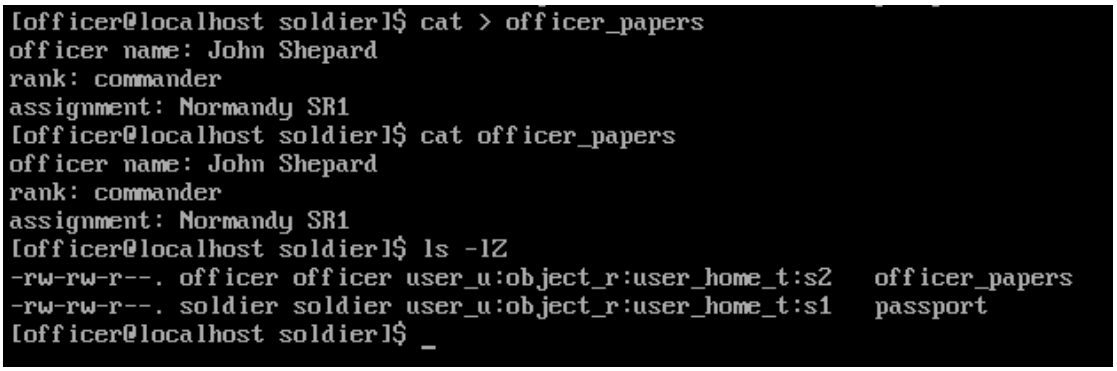
Давайте проверим:

Пусть пользователь солдат создаст файл «паспорт». Как можно видеть в классической DAC, содержимое этого файла могут читать другие пользователи.

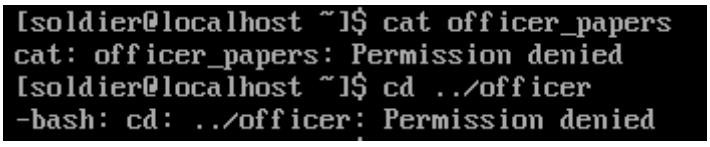
```
NauLinux QNet 7.7 (Nitrogen)
Kernel 3.10.0-1160.15.2.el7.x86_64 on an x86_64

localhost login: soldier
Password:
[soldier@localhost ~]$ ls
[soldier@localhost ~]$ cat > passport
soldier name: Kaiden Alenko
rank: 1st sergeant[soldier@localhost ~]$
[soldier@localhost ~]$ cat passport
soldier name: Kaiden Alenko
rank: 1st sergeant[soldier@localhost ~]$ cat >> passport

[soldier@localhost ~]$ cat passport
soldier name: Kaiden Alenko
rank: 1st sergeant
[soldier@localhost ~]$ id -Z
user_u:user_r:user_t:s1
[soldier@localhost ~]$ ls -lZ
-rw-rw-r--. soldier soldier user_u:object_r:user_home_t:s1   passport
[soldier@localhost ~]$ _
```

```
[officer@localhost home]$ ls
officer  solar  soldier  user  usertest1
[officer@localhost home]$ id -Z
user_u:user_r:user_t:s2
[officer@localhost home]$ cd soldier
[officer@localhost soldier]$ ls
passport
[officer@localhost soldier]$ cat passport
soldier name: Kaiden Alenko
rank: 1st sergeant
```

Пользователь офицер спокойно читает «паспорт» солдата.



Однако, офицер оставляет у солдата свои «бумаги». Как можно видеть в классической DAC, содержимое этого файла могут читать другие пользователи.



Солдат не может ни прочитать содержимое файла, ни зайти в домашнюю директорию офицера, хотя root расставил на домашние папки разрешения 777.

Таким образом мы можем видеть, что мы смогли настроить двухуровневую модель безопасности и заставить ее работать.

Задание 3. PAM модуль

Наш модуль будет типа auth и обязательности required, применять его будем в качестве двухфакторной аутентификации в login. Что будет делать модуль: отправлять на почтовый ящик google токен авторизации, который необходимо будет ввести в промпте логина после ввода основного пароля пользователя linux (в конкретно нашем примере мы не стали разворачивать локальный мини SMTP сервер, как и не стали писать мини базу имя_пользователя->email, однако при желании это можно сделать). Для отправки писем используем libcurl (библиотека утилиты curl для Си), в настройках опций libcurl в коде модуля авторизуемся почтой и паролем gmail (можно использовать пароль приложений, который генерирует гугл при запросе в настройках учетной записи), после чего посылаем запрос на smtp-сервер гугл “smtps://smtp.gmail.com:465”, который в свою очередь доставляет сообщение до адресата. Очевидно, в коде модуля будет видно, что я, по сути, сам посылаю из модуля сообщение себе, но я могу послать и сообщение на любой другой ящик, который, например, будет работать в режиме пересылки (например, по сотрудникам) и т.д. и т.п..
Код модуля:

pam_email.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <syslog.h>
#include <time.h>
#include <curl/curl.h>
#include <curl/easy.h>

#define PAM_SM_AUTH

#include <security/pam_appl.h>
#include <security/pam_modules.h>
#include <security/_pam_macros.h>

#define FROM_ADDR    "<egor.dom0923@gmail.com>"
#define TO_ADDR      "<egor.dom0923@gmail.com>"

#define FROM_MAIL "From_ADMIN" FROM_ADDR
#define TO_MAIL   "nice_token" TO_ADDR
#define TOKSYMBOLS "123456789abcdefghijkmnopqrstuvwxyzABCDEFGHJKLMNPQRSTUVWXYZ"
#define TOKLEN 12

struct upload_status {
  int lines_read;
};

static const char *payload_text[] = {
  "Date: Mon, 29 Nov 2010 21:54:29 +1100\r\n",
  "To: " TO_MAIL "\r\n",
  "From: " FROM_MAIL "\r\n",
  "Message-ID: <dcd7cb36-11db-487a-9f3a-e652a9458efd@"
  "rfcpedant.example.org>\r\n",
  "Subject: System auth token\r\n",
  "\r\n", /* empty line to divide headers from body, see RFC5322 */
  "Your token is:\r\n",
  "\r\n",
  "Have a nice auth.\r\n",
  "Behave accordingly!\r\n",
  NULL
};

char* time_now(){
    int RFC1123_TIME_LEN = 29;
    static const char* DAY_NAMES  [] = { "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat" };
    static const char* MONTH_NAMES[] = { "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" };
    char* ret = calloc(RFC1123_TIME_LEN + 1, sizeof(char));
    time_t t;
    struct tm tmm;
    time(&t);
    gmtime_r(&t, &tmm);
    strftime(&ret[0], RFC1123_TIME_LEN + 1, "---, %d --- %Y %H:%M:%S GMT", &tmm);
    memcpy(&ret[0], DAY_NAMES[tmm.tm_wday], 3);
    memcpy(&ret[8], MONTH_NAMES[tmm.tm_mon],  3);

    return ret;
}
```

```c
char* generate_token(int length){
    char *values = TOKSYMBOLS;
    int mod = strlen(values);
    char *retval = calloc(length + 1, sizeof(char));
    int i;

    srand(time(NULL));
    for(i = 0; i < length; i++)
        retval[i] = values[rand() % mod];

    return retval;
}

char* gen_message_id(){
    char* from_part = "@gmail.com>\r\n";
    int id_len = 37;
    char* id = generate_token(id_len);
    char* ret = calloc(strlen("Message-ID: ") + id_len + strlen(from_part) + 1, sizeof(char));
    strcpy(ret, "Message-ID: ");
    strcat(ret, id);
    strcat(ret, from_part);

    return ret;
}


static size_t payload_source(char *ptr, size_t size, size_t nmemb, void *userp)
{
    struct upload_status *upload_ctx = (struct upload_status *)userp;
    const char *data;

    if((size == 0) || (nmemb == 0) || ((size*nmemb) < 1)) {
        return 0;
    }

    data = payload_text[upload_ctx->lines_read];

    if(data) {
        size_t len = strlen(data);
        memcpy(ptr, data, len);
        upload_ctx->lines_read++;

        return len;
    }

    return 0;
}
```

```
int pam_converse(pam_handle_t *pamh, const char *message, char **response, int type) {
    int pam_err = 0;

    char *mresponse = NULL;

    struct pam_conv *conv;
    struct pam_message msg;
    const struct pam_message *msgp;
    struct pam_response *resp;

    pam_err = pam_get_item(pamh, PAM_CONV, (const void **) &conv);

    if (pam_err != PAM_SUCCESS)
        return -1;

    msg.msg_style = type;
    msg.msg = message;
    msgp = &msg;

    resp = NULL;
    pam_err = (*conv->conv)(1, &msgp, &resp, conv->appdata_ptr);

    if (resp != NULL) {
        if (pam_err == PAM_SUCCESS) {
            mresponse = resp->resp;
            pam_err = 0;
        }
        else {
            free(resp->resp);
            pam_err = -1;
        }
        free(resp);
    }

    response[0] = mresponse;
    return pam_err;
}
```

```
PAM_EXTERN int pam_sm_authenticate(pam_handle_t *pamh, int flags, int argc, const char **argv) {
    CURLcode res = CURLE_OK;
    struct curl_slist *recipients = NULL;

    struct upload_status upload_ctx;

    upload_ctx.lines_read = 0;
    char* token = NULL;
    char* recv_token = NULL;
    char* dattime = NULL;
    char* mesID = NULL;
    int retval;
    const char *user = NULL;
    int num_char_tok = TOKLEN;


    retval = pam_get_user(pamh, &user, NULL);
    if (retval != PAM_SUCCESS) {
        syslog(LOG_ALERT, "get user returned error: %s", pam_strerror(pamh, retval));
        return retval;
    }

    if (user == NULL || *user == '\0') {
        return PAM_USER_UNKNOWN;
    }
```

(далее на след стр.)

```
    CURL *curl = curl_easy_init();
    if (curl) {
        curl_easy_setopt(curl, CURLOPT_VERBOSE, 1);

        //curl_easy_setopt(curl, CURLOPT_URL, "imap://imap.gmail.com:993");
        curl_easy_setopt(curl, CURLOPT_URL, "smtps://smtp.gmail.com:465"); //587

        curl_easy_setopt(curl, CURLOPT_USERNAME, "egor.dom0923@gmail.com");
        curl_easy_setopt(curl, CURLOPT_PASSWORD, "[redacted]");

        curl_easy_setopt(curl, CURLOPT_USE_SSL, (long) CURLUSESSL_ALL);

        curl_easy_setopt(curl, CURLOPT_MAIL_FROM, FROM_ADDR);

        recipients = curl_slist_append(recipients, TO_ADDR);
        curl_easy_setopt(curl, CURLOPT_MAIL_RCPT, recipients);

        token = generate_token(TOKLEN);
        char* body = malloc(strlen(token) + 1 + 2);
        strcpy(body, token); /* copy name into the new var */
        strcat(body, "\r\n"); /* add the extension */
        payload_text[7] = (const char*)body;

        dattime = time_now();
        char* DATE = malloc(strlen(dattime) + strlen("Date: ") + 1 + 2);
        strcpy(DATE, "Date: ");
        strcat(DATE, dattime);
        strcat(DATE, "\r\n");
        payload_text[0] = (const char*)DATE;

        mesID = gen_message_id();
        payload_text[4] = (const char*)mesID;


        curl_easy_setopt(curl, CURLOPT_READFUNCTION, payload_source);
        curl_easy_setopt(curl, CURLOPT_READDATA, &upload_ctx);
        curl_easy_setopt(curl, CURLOPT_UPLOAD, 1L);

        /* Send the message */
        res = curl_easy_perform(curl);

        /* Check for errors */
        if (res != CURLE_OK)
            fprintf(stderr, "curl_easy_perform() failed: %s\n",
                curl_easy_strerror(res));

        curl_slist_free_all(recipients);
        curl_easy_cleanup(curl);
    }
```

```
    if (pam_converse(pamh, "Token:", &recv_token, PAM_PROMPT_ECHO_ON) != 0) {
        free(token);
        return PAM_CONV_ERR;
    }

    // compare the sent token and the received one and respond correspondingly
    if (strncmp(token, recv_token, num_char_tok) == 0) {
        retval = PAM_SUCCESS;
    }
    else {
        retval = PAM_AUTH_ERR;
    }

    recv_token = NULL;
    free(token);
    free(mesID);
    free(dattime);
    user = NULL;
    return retval;


}

PAM_EXTERN int pam_sm_setcred(pam_handle_t *pamh, int flags, int argc, const char **argv){
    return PAM_SUCCESS;
}


#ifdef PAM_STATIC


struct pam_module _pam_permit_modstruct = {
    "pam_email",
    pam_sm_authenticate,
    pam_sm_setcred,
    NULL,
    NULL,
    NULL,
    NULL
};

#endif
```

Теперь скомпилируем в shared object и закинем в папку /usr/lib/x86_64-linux-gnu/security:

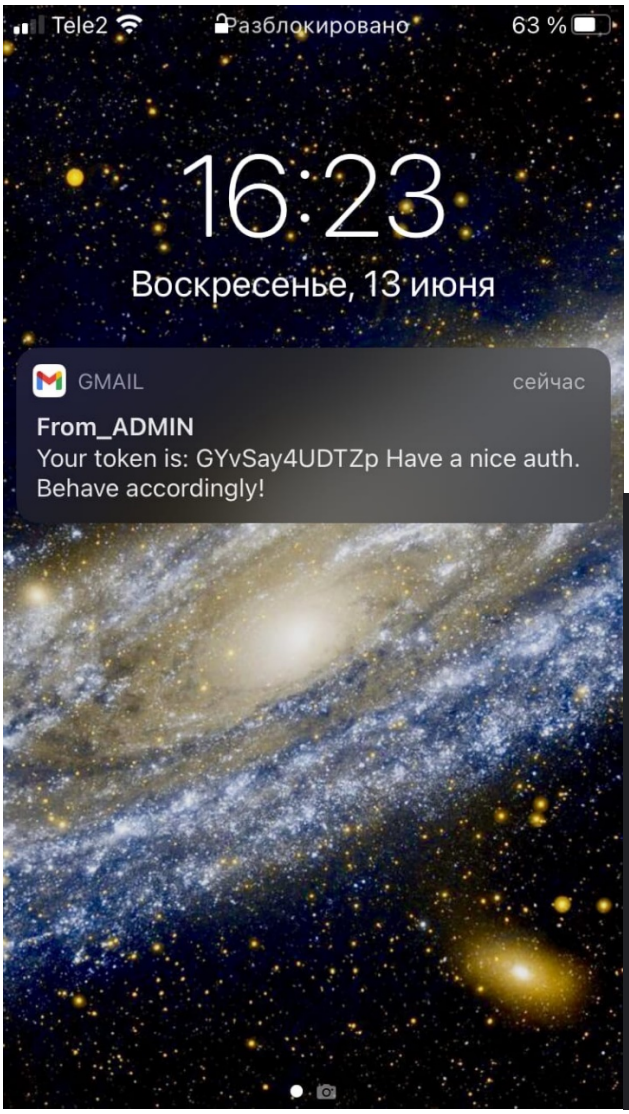
```
root@kali:/mnt/hgfs/vm_share/labs/OS/lab8# gcc pam_email.c -o pam_email.so -lcurl -shared -fPIC
root@kali:/mnt/hgfs/vm_share/labs/OS/lab8# cp pam_email.so /usr/lib/x86_64-linux-gnu/security/
```

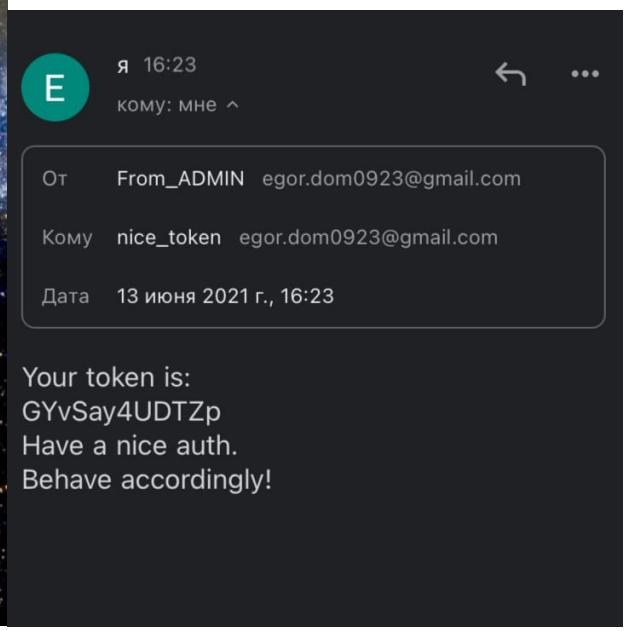
Теперь давайте изменим конфиг файл для login и добавим строчку с указанием использования нашего модуля:

```
# Standard Un*x authentication.
@include common-auth
auth required pam_email.so
# This allows certain extra groups to be granted to a user
```

Готово, давайте протестируем:

Вуаля.






* subjectAltName: host "smtp.gmail.com" matched cert's "smtp.gmail.com"
* issuer: C=US; O=Google Trust Services; CN=GTS CA 1O1
* SSL certificate verify ok.
* old SSL session ID is stale, removing
< 220 smtp.gmail.com ESMTP x15sm1180104lfa.156 - gsmtp
> EHLO kali
< 250-smtp.gmail.com at your service, [83.102.217.208]
< 250-SIZE 35882577
< 250-8BITMIME
< 250-AUTH LOGIN PLAIN XOAUTH2 PLAIN-CLIENTTOKEN OAUTHBEARER XOAUTH
< 250-ENHANCEDSTATUSCODES
< 250-PIPELINING
< 250-CHUNKING
< 250 SMTPUTF8
> AUTH PLAIN
< 334
> AGVnb3IuZG9tMDkyM0BnbWFpbC5jb20AbnFsZnR4Z3phb3hob2Z2l2w==
< 235 2.7.0 Accepted
> MAIL FROM:<egor.dom0923@gmail.com>
< 250 2.1.0 OK x15sm1180104lfa.156 - gsmtp
> RCPT TO:<egor.dom0923@gmail.com>
< 250 2.1.5 OK x15sm1180104lfa.156 - gsmtp
> DATA
< 354 Go ahead x15sm1180104lfa.156 - gsmtp
< 250 2.0.0 OK 1623590629 x15sm1180104lfa.156 - gsmtp
* Connection #0 to host smtp.gmail.com left intact
Token:GYoSay4UDTZp
Linux kali 5.10.28-custom #1 SMP Fri Apr 23 02:34:27 MSK 2021 x86_64
The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Jun 13 00:15:18 MSK 2021 on tty2
test@kali:~$ _